Classification de textes à l'aide de pyspark ML (Disaster Tweets)

Master Sciences des Données et Systèmes Intelligents

Abdelhafid AYAOU Mounir DOUIRI

March 20, 2022

1 Classification de textes à l'aide de pyspark ML

1.1 Introduction

Le traitement du langage naturel NLP est l'un des processus importants pour les équipes de science des données à travers le monde. Avec des données en constante augmentation, la plupart des organisations ont déjà migré vers des plateformes de Big Data comme Apache Hadoop et des offres cloud comme AWS, Azure et GCP. Ces plates-formes sont plus que capables de gérer le Big Data, ce qui permet aux organisations d'effectuer des analyses à grande échelle pour des données non structurées telles que la catégorisation de texte. Mais en matière d'apprentissage automatique, il existe toujours un fossé entre les systèmes de mégadonnées et les outils d'apprentissage automatique.

Les bibliothèques python d'apprentissage automatique populaires telles que scikit-learn et Gensim sont hautementoptimisées pour fonctionner sur des machines à nœud unique et ne sont pas conçues pour les environnements distribués. **Apache Spark MLlib** est l'un des nombreux outils qui aident à combler cette lacune en offrant la plupart des modèles d'apprentissage automatique comme la régression linéaire, la régression logistique, SVM, Random Forest, K-means, SVM et bien d'autres pour effectuer les tâches d'apprentissage automatique les plus courantes. Le Traitement Automatique du Langage naturel (TAL) ou Natural Language Processing (NLP) en anglais trouve de nombreuses applications dans la vie de tous les jours:

- Traduction de texte
- Correcteur orthographique
- Résumé automatique d'un contenu
- Synthèse vocale
- Analyse d'opinion/sentiment
- Prédiction du prochain mot sur smartphone
- Extraction des entités nommées depuis un texte ...

1.2 Qu'est-ce que Spark NLP ?

Spark NLP est une bibliothèque open-source, lancée il y a un peu plus de 4 ans, dans le but de fournir un NLP de pointe à la communauté open-source, offrant des bibliothèques et des API complètes en Python, Java et Scala. Évolué à la suite de la croissance de l'apprentissage en profondeur dans les technologies NLP et de l'optimisation d'Apache Spark, il permet de faire fonctionner les choses d'un ou deux ordres de grandeur plus rapidement sur le même matériel pour les bibliothèques basées sur Spark.

De plus, il est construit directement sur Spark ML, ce qui signifie qu'un pipeline Spark NLP est de la même classe qu'une construction de pipeline Spark ML, offrant ainsi une série d'avantages.

1.3 Description de problèmatique :

Twitter est devenu un canal de communication important en cas d'urgence.

L'omniprésence des smartphones permet aux gens d'annoncer une urgence qu'ils observent en temps réel. Pour cette raison, de plus en plus d'agences sont intéressées par la surveillance programmatique de Twitter (c'est-à-dire les organisations de secours en cas de catastrophe et les agences de presse).Mais, il n'est pas toujours clair si les mots d'une personne annoncent réellement une catastrophe.

Dans cette compétition, nous sommes mis au défi de construire un modèle d'apprentissage automatique qui prédit quels Tweets concernent de vraies catastrophes et lesquels ne le sont pas. Nous aurons accès à un ensemble de données de 10 000 tweets classés à la main: Link of Dataset

1.4 Qu'est-ce qu'on prédit ?

Nous prédisons si un tweet donné concerne une véritable catastrophe ou non. Si oui, prédisez un 1. Si non, prédisez un 0.

1.5 Description des données

Les colonnes incluses dans cet ensemble de données sont :

- **id:** Cette colonne comprend des identifiants pour chaque tweets
- **keyword:** Keyword de ce tweet (peut être vide)
- **location:** Lieu d'où le tweet a été envoyé (peut aussi être vide)
- **text:** Le texte d'un tweet
- **target:** Dans le fichier train.csv uniquement, cela indique si un tweet concerne une véritable catastrophe (1) ou non (0)

2 la création de session spark et l'importation des données

2.1 Importation des bibliothèques nécessaires

[1]:
```
import numpy as np
import pandas as pd
from pyspark.sql import SparkSession
from pyspark.ml.feature import CountVectorizer,StringIndexer,␣
 ↪RegexTokenizer,StopWordsRemover
from pyspark.sql.functions import col,regexp_replace
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

2.2 Création d'une session spark

La création d'une session spark est le point d'entrée de Spark ML. C'est l'un des tout premiers objets que nous créons en réalisant un projet Spark ML. nous créons un SparkSession en utilisant la méthode *SparkSession.builder* (qui nous donne accès à l'API Builder que nous utilisons pour configurer la session).

[2]:
```
spark = SparkSession.builder \
    .master("local[*]") \
    .config("spark.executor.memory", "8g") \
    .config("spark.driver.memory", "8g") \
    .appName("NLP") \
    .getOrCreate()
```

2.3 Importation de Dataset

[4]:
```
dataset = spark.read.csv('DisasterTweets.csv', inferSchema=True, header=True)
dataset.printSchema()
```

```
root
 |-- id: string (nullable = true)
 |-- keyword: string (nullable = true)
 |-- location: string (nullable = true)
 |-- text: string (nullable = true)
 |-- target: integer (nullable = true)
```

Predictor variables : id, keyword, location, text

Outcome variable : target

Voyons à quoi ressemblent les données: Pandas data frame est meilleur que la fonction Spark DataFrame *show()* , pour cela nous utiliserons Pandas DataFrame.

[5]:
```
dataset.toPandas().head()
```

[5]:
```
   id keyword location                                              text  \
0   1    None     None  Our Deeds are the Reason of this #earthquake M...
1   4    None     None             Forest fire near La Ronge Sask. Canada
2   5    None     None  All residents asked to 'shelter in place' are ...
3   6    None     None  13,000 people receive #wildfires evacuation or...
4   7    None     None  Just got sent this photo from Ruby #Alaska as ...

   target
0     1.0
1     1.0
2     1.0
3     1.0
4     1.0
```

[6]:
```
print("Data Record Count:",dataset.count())
```

```
Data Record Count: 8387
```

Visualisation de **'target'** (Outcome column)

[7]:
```
dataset.toPandas().groupby(['target']).size()
```

[7]:
```
target
0.0    4095
1.0    3081
dtype: int64
```

Alors Les données sont bien équilibrées.

3 Prétraitement des données

Une étape essentielle dans la fouille des données textuelles est la préparation des données collectées afin qu'elles puissent être efficacement traitées par les algorithmes d'apprentissage machine. Il contient de deux parties **La partie de nettoyage des donnees** qui consiste en un certain nombre de prétraitements qui visent à standardiser et nettoyer le contenu des tweets qui sont archivés :

- Suppression des valeurs nulles
- Suppression des chiffres dans les tweets
- Séparation des mots du tweet
- Suppression des mots vides de sens
- Vectorisation

Et la deuxième partie de **séparation des données en train et validation data.**

Pré-traitement est une étape qui cherche à standardiser du texte afin de rendre son usage plus facile. C'est le processus de conversion des données en quelque chose qu'un ordinateur peut comprendre.

[8]:
```
dataset = dataset.select("id","text","target")
dataset.toPandas().head()
```

[8]:

	id	text	target
0	1	Our Deeds are the Reason of this #earthquake M...	1.0
1	4	Forest fire near La Ronge Sask. Canada	1.0
2	5	All residents asked to 'shelter in place' are ...	1.0
3	6	13,000 people receive #wildfires evacuation or...	1.0
4	7	Just got sent this photo from Ruby #Alaska as ...	1.0

3.1 Nettoyage des donnees

Le nettoyage des données est un processus qui vise à identifier et corriger les données altérées, inexactes ou non pertinentes. Cette étape fondamentale du traitement des données améliore la cohérence, fiabilité et valeur des données.

3.2 Suppression des valeurs nulles

drop : est une fonction utilisée pour supprimer des lignes avec des valeurs NULL dans les colonnes de notre DataFrame

[9]:
```
dataset = dataset.dropna()
print("Data Record Count:",dataset.count())
```

```
Data Record Count: 7176
```

3.2.1 Suppression des chiffres dans les tweets

Les textes des tweets contiennent de nombreux chiffres qui nous pouvons les supprimer on utilise la fonction Spark **regexp_replace**, qui permet en général de remplacer une valeur de colonne par une chaîne pour une autre chaîne/sous-chaîne.

[10]:
```
dataset = dataset.withColumn("only_str",regexp_replace(col('text'), '\d+', ''))
dataset.toPandas().head()
```

[10]:

	id	text	target	only_str
0	1	Our Deeds are the Reason of this #earthquake M...	1	Our Deeds are the Reason of this #earthquake M...
1	4	Forest fire near La Ronge Sask. Canada	1	Forest fire near La Ronge Sask. Canada
2	5	All residents asked to 'shelter in place' are ...	1	All residents asked to 'shelter in place' are ...
3	6	13,000 people receive #wildfires evacuation or...	1	, people receive #wildfires evacuation orders ...
4	7	Just got sent this photo from Ruby #Alaska as ...	1	Just got sent this photo from Ruby #Alaska as ...

3.2.2 Séparation des mots du tweet

Le but de ce prétraitement est d'extraire d'un texte brut les différents mots le composant afin de pouvoir les analyser séparément. Cette étape est un préalable pour de nombreux traitements comme la lemmatisation. nous avons utilisé la fonction Spark **RegexTokenizer** pour cette étape particulière. c'est une bonne option pour le faire.

[11]:
```
regex_tokenizer = RegexTokenizer(inputCol="only_str", outputCol="words",␣
 ↪pattern="\\W")
raw_words = regex_tokenizer.transform(dataset)
raw_words.toPandas().head()
```

[11]:
```
   id                                               text  target  \
0   1  Our Deeds are the Reason of this #earthquake M...       1
1   4             Forest fire near La Ronge Sask. Canada       1
2   5  All residents asked to 'shelter in place' are ...       1
3   6  13,000 people receive #wildfires evacuation or...       1
4   7  Just got sent this photo from Ruby #Alaska as ...       1

                                            only_str  \
0  Our Deeds are the Reason of this #earthquake M...
1              Forest fire near La Ronge Sask. Canada
2  All residents asked to 'shelter in place' are ...
3  , people receive #wildfires evacuation orders ...
4  Just got sent this photo from Ruby #Alaska as ...

                                               words
0  [our, deeds, are, the, reason, of, this, earth...
1      [forest, fire, near, la, ronge, sask, canada]
2  [all, residents, asked, to, shelter, in, place...
3  [people, receive, wildfires, evacuation, order...
4  [just, got, sent, this, photo, from, ruby, ala...
```

3.2.3 Suppression des mots vides de sens

Le but de ce prétraitement est de supprimer les mots qui n'ont pas de signification particulière comme des conjonctions (ex : and, are, our, ...) et qui peuvent être ignorés.

nous avons utilisé la fonction Spark **StopWordsRemover** pour supprimer ces mots vides de sens (stopWords) que les mots apparaissent fréquemment et n'ont pas autant de sens. Spark **StopWordsRemover** prend en entrée une séquence de chaînes. Ensuite, il supprime tous les mots vides des séquences d'entrée.

[12]:
```
remover = StopWordsRemover(inputCol="words", outputCol="filtered")
words_df = remover.transform(raw_words)
words_df.select("id","words","target","filtered").toPandas().head()
```

[12]:
```
   id                                               words  target  \
0   1  [our, deeds, are, the, reason, of, this, earth...       1
```

```
1  4      [forest, fire, near, la, ronge, sask, canada]       1
2  5  [all, residents, asked, to, shelter, in, place...       1
3  6  [people, receive, wildfires, evacuation, order...       1
4  7  [just, got, sent, this, photo, from, ruby, ala...       1

                                              filtered
0  [deeds, reason, earthquake, may, allah, forgiv...
1      [forest, fire, near, la, ronge, sask, canada]
2  [residents, asked, shelter, place, notified, o...
3  [people, receive, wildfires, evacuation, order...
4  [got, sent, photo, ruby, alaska, smoke, wildfi...
```

3.2.4 Vectorisation

La représentation vectorielle des données vise à transformer l'ensemble des textes des tweets sous la forme de vecteurs afin qu'ils puissent être traités par les méthodes de fouille conçues pour des données vectorielles.

CountVectorizer : fonctionnent sur le nombre de mots (jetons). Il utilise des mots dans des documents texte pour créer des vecteurs contenant le nombre de jetons. Il prévoit l'utilisation d'un dictionnaire de mots pour identifier les jetons qui peuventêtre pris en entrée d'algorithmes. CountVectorizer compte les fréquences de mots pour le document, c'est un outil utilisé pour convertir le texte en vecteur.

[50]:
```
cv = CountVectorizer(inputCol="filtered", outputCol="features")
model = cv.fit(words_df)
countVectorizer_train = model.transform(words_df)
countVectorizer_train = countVectorizer_train.withColumn("label",col('target'))
pd.set_option("expand_frame_repr", True)
pd.set_option("colheader_justify", "left")

countVectorizer_train.toPandas().head()
```

[50]:
```
  id text                                                target  \
0  1  Our Deeds are the Reason of this #earthquake M...  1
1  4             Forest fire near La Ronge Sask. Canada  1
2  5  All residents asked to 'shelter in place' are ...  1
3  6  13,000 people receive #wildfires evacuation or...  1
4  7  Just got sent this photo from Ruby #Alaska as ...  1

  only_str                                             \
0  Our Deeds are the Reason of this #earthquake M...
1             Forest fire near La Ronge Sask. Canada
2  All residents asked to 'shelter in place' are ...
3  , people receive #wildfires evacuation orders ...
4  Just got sent this photo from Ruby #Alaska as ...

  words                                                \
```

```
0  [our, deeds, are, the, reason, of, this, earth...
1      [forest, fire, near, la, ronge, sask, canada]
2  [all, residents, asked, to, shelter, in, place...
3  [people, receive, wildfires, evacuation, order...
4  [just, got, sent, this, photo, from, ruby, ala...

  filtered                                                  \
0  [deeds, reason, earthquake, may, allah, forgiv...
1      [forest, fire, near, la, ronge, sask, canada]
2  [residents, asked, shelter, place, notified, o...
3  [people, receive, wildfires, evacuation, order...
4  [got, sent, photo, ruby, alaska, smoke, wildfi...

  features                                                label
0  (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...  1
1  (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, ...  1
2  (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...  1
3  (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...  1
4  (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...  1
```

voici la résultat de notre prétraitement des données

[51]:
```
countVectorizer_train.select('text','words','filtered','features','target').
↪toPandas().head()
```

[51]:
```
  text                                                    \
0  Our Deeds are the Reason of this #earthquake M...
1              Forest fire near La Ronge Sask. Canada
2  All residents asked to 'shelter in place' are ...
3  13,000 people receive #wildfires evacuation or...
4  Just got sent this photo from Ruby #Alaska as ...

  words                                                   \
0  [our, deeds, are, the, reason, of, this, earth...
1      [forest, fire, near, la, ronge, sask, canada]
2  [all, residents, asked, to, shelter, in, place...
3  [people, receive, wildfires, evacuation, order...
4  [just, got, sent, this, photo, from, ruby, ala...

  filtered                                                \
0  [deeds, reason, earthquake, may, allah, forgiv...
1      [forest, fire, near, la, ronge, sask, canada]
2  [residents, asked, shelter, place, notified, o...
3  [people, receive, wildfires, evacuation, order...
4  [got, sent, photo, ruby, alaska, smoke, wildfi...

  features                                                target
```

```
0  (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...  1
1  (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, ...  1
2  (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...  1
3  (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...  1
4  (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...  1
```

3.3 Séparation des données en train et validation data

Cette étape consiste en la création de deux jeux de données qui seront utilisés pour l'entrainement et la mesure de performance des différents modèles sélectionnés. Pour maximiser la performance des modèles prédictifs, nous avons opté pour une séparation aléatoire des tweets avec une proportion de 80% des données réservées à la phase d'entrainement et de 20% des données réservées pour la mesure de performance des modèles.

- L'ensemble de données d'entrainement: **80%**
- L'ensemble de données pour le test: **20%**

Pour cela **randomSplit** nous permet de diviser de manière transparente les données en ensembles d'entraînement et de test.

[15]:
```
(train, validate) = countVectorizer_train.randomSplit([0.8, 0.2], seed = 123)
```

4 Machine Learning : Modèles de prédiction

Nous avons préparé notre ensemble de données et les diviser en ensembles d'apprentissage (training) et de test (testing). Il est maintenant temps de choix du modèle d'analyse.
Choisir un modèle revient à choisir l'algorithme à utiliser pour construire le modèle. Il existe de nombreux algorithmes d'apprentissage machine et plusieurs d'entre eux seront testés et comparés afin de trouver le meilleur modèle.

Ces algorithmes peuvent être catégorisés selon le mode d'apprentissage et le type du problème traité :

- Apprentissage supervisé (supervised learning)
- Apprentissage non supervisé (unsupervised learning)
- Apprentissage par renforcement (reinforcement learning)

Pour ce projet, le choix s'est porté sur l'apprentissage supervisé qui est particulièrement bien adapté à la prédiction de résultats en s'aidant d'une base de données déjà étiquetés.

On distingue ensuite au niveau de l'apprentissage supervisé deux types d'algorithmes, selon que la variable à prédire est quantitative ou qualitative :

- **Les algorithmes de régression :** Ils sont adaptés aux problèmes pour lesquels le résultat à prédire est une valeur numérique.
- **Les algorithmes de classification :** Ils tentent d'affecter une étiquette à une observation faite sur un ensemble de variables. L'ensemble des valeurs possibles pour cette étiquette doit être dénombrable et défini au préalable.

Pour ce projet, la variable de sortie est qualitative (tweet concerne une véritable catastrophe, non concerne une véritable catastrophe). nous avons donc choisi de tester des algorithmes de classification parmi ceux qui sont disponibles dans la librairie Spark MLlib afin de bénéficier du passage à l'échelle de Spark. Le choix s'est porté sur les algorithmes suivants qui sont compatibles avec la classification binaire et qui sont décrits brièvement dans la suivante:

- Naïves Bayes
- Régression Logistique
- Arbres de Décision
- Forêts Aléatoires (Random Forest)
- Gradient Boosting
- SVM Support vector machine

Il est rare qu'un seul modèle suffise à produire de bonnes prédictions pour l'ensemble des données. Nous avons donc testé la combinaison de plusieurs modèles qui permet souvent d'améliorer la performance globale de prédiction des modèles sous-jacents.

4.1 Les grandes étapes pour chacun algorithmes:

4.1.1 Entrainement des modèles

Cette étape a pour objectif d'appliquer un modèle d'apprentissage aux données d'entrainement selon le processus suivant :

- **Définition d'un pipeline pour le modèle concerné :** Spark permet de définir des pipelines qui sont des flux de travail qui enchaînent différentes étapes de traitement des données se trouvant dans des DataFrames.
- **Définition des hyperparamètres du modèle :** Chaque algorithme d'apprentissage comporte un certain nombre de paramètres qui permettent d'optimiser la performance du modèle sélectionné.
- **Entrainement et optimisation du modèle :** L'apprentissage du modèle sur le jeu de données d'entrainement s'effectue par la fonction Spark fit avec en paramètres le jeu de données concerné.

4.1.2 Prédictions et évaluations des Résultats

Cette étape consiste à évaluer chacun des modèles sur les même données de validation afin de trouver le modèle ayant les meilleures performances pour la prédiction si un tweet donné concerne une véritable catastrophe ou non.

- **Prédiction des modèles :** Une fois qu'un modèle est entrainé, il est possible d'appliquer ce modèle sur de nouvelles données en utilisant la fonction Spark *transform()*. Les prédictions obtenues sur le jeu de validation permettent de mesurer la performance de généralisation de chacun des modèles.
- **Evaluation des modèles :** L'évaluation d'un modèle consiste à mesurer les écarts entre les prédictions du modèle et les résultats attendus. Spark propose les mesures suivantes pour

les algorithmes de classification: **Precision - Recall - Accuracy**

Ce qui nous importe, c'est l'accuracy par ce qu'est une mesure qui représente le pourcentage de prédictions correctes de l'algorithme.

4.1.3 Naïves Bayes

Entrainement de modèle

[18]:
```
from pyspark.ml.classification import NaiveBayes

nb = NaiveBayes(modelType="multinomial", labelCol="target",␣
 ↪featuresCol="features")
nbModel = nb.fit(train)
```

Prédiction et évaluation de Résultat

[19]:
```
nbPreds = nbModel.transform(validate)
evaluator = MulticlassClassificationEvaluator(labelCol="target",␣
 ↪predictionCol="prediction", metricName="accuracy")
nb_accuracy = evaluator.evaluate(nbPreds)
print("Accuracy of NaiveBayes is = %g"% (nb_accuracy))
```

```
Accuracy of NaiveBayes is = 0.796832
```

4.1.4 Régression Logistique

Entrainement de modèle

[21]:
```
from pyspark.ml.classification import LogisticRegression

lr = LogisticRegression(featuresCol = 'features', labelCol = 'target',␣
 ↪maxIter=10)
lrModel = lr.fit(train)
```

Prédiction et évaluation de Résultat

[22]:
```
lrPreds = lrModel.transform(validate)
evaluator = MulticlassClassificationEvaluator(labelCol="target",␣
 ↪predictionCol="prediction", metricName="accuracy")
lr_accuracy = evaluator.evaluate(lrPreds)
print("Accuracy of Logistic Regression is = %g"% (lr_accuracy))
```

```
Accuracy of Logistic Regression is = 0.772727
```

4.1.5 Arbres de Décision

Entrainement de modèle

[23]:
```
from pyspark.ml.classification import DecisionTreeClassifier

dt = DecisionTreeClassifier(featuresCol = 'features', labelCol = 'target',␣
 ↪maxDepth = 3)
dtModel = dt.fit(train)
```

Prédiction et évaluation de Résultat

[24]:
```
dtPreds = dtModel.transform(validate)
evaluator = MulticlassClassificationEvaluator(labelCol="target",␣
 ↪predictionCol="prediction", metricName="accuracy")
dt_accuracy = evaluator.evaluate(dtPreds)
print("Accuracy of Decision Trees is = %g"% (dt_accuracy))
```

```
Accuracy of Decision Trees is = 0.652204
```

4.1.6 Forêts Aléatoires (Random Forest)

Entrainement de modèle

[25]:
```
from pyspark.ml.classification import RandomForestClassifier

rf = RandomForestClassifier(featuresCol = 'features', labelCol = 'target')
rfModel = rf.fit(train)
```

Prédiction et évaluation de Résultat

[26]:
```
rfPreds = rfModel.transform(validate)
evaluator = MulticlassClassificationEvaluator(labelCol="target",␣
 ↪predictionCol="prediction", metricName="accuracy")
rf_accuracy = evaluator.evaluate(rfPreds)
print("Accuracy of Random Forests is = %g"% (rf_accuracy))
```

```
Accuracy of Random Forests is = 0.616391
```

4.1.7 Gradient Boosting

Entrainement de modèle

[27]:
```
from pyspark.ml.classification import GBTClassifier

gbt = GBTClassifier(maxIter=10)
gbtModel = gbt.fit(train)
```

Prédiction et évaluation de Résultat

[28]:
```
gbtPreds = gbtModel.transform(validate)
evaluator = MulticlassClassificationEvaluator(labelCol="target",␣
 ↪predictionCol="prediction", metricName="accuracy")
gb_accuracy = evaluator.evaluate(gbtPreds)
```

```
print("Accuracy of GBT is = %g"% (gb_accuracy))
```

```
Accuracy of GBT is = 0.674931
```

4.1.8 SVM Support vector machine

Entrainement de modèle

[29]:
```
from pyspark.ml.classification import LinearSVC

svm = LinearSVC(maxIter=10, regParam=0.1, featuresCol = 'features', labelCol =␣
 ↪'target')
svmModel = rf.fit(train)
```

Prédiction et évaluation de Résultat

[30]:
```
svmPreds = rfModel.transform(validate)
evaluator = MulticlassClassificationEvaluator(labelCol="target",␣
 ↪predictionCol="prediction", metricName="accuracy")
svm_accuracy = evaluator.evaluate(svmPreds)
print("Accuracy of SVM is = %g"% (svm_accuracy))
```

```
Accuracy of SVM is = 0.616391
```

5 Comparaison de la performance des modèles

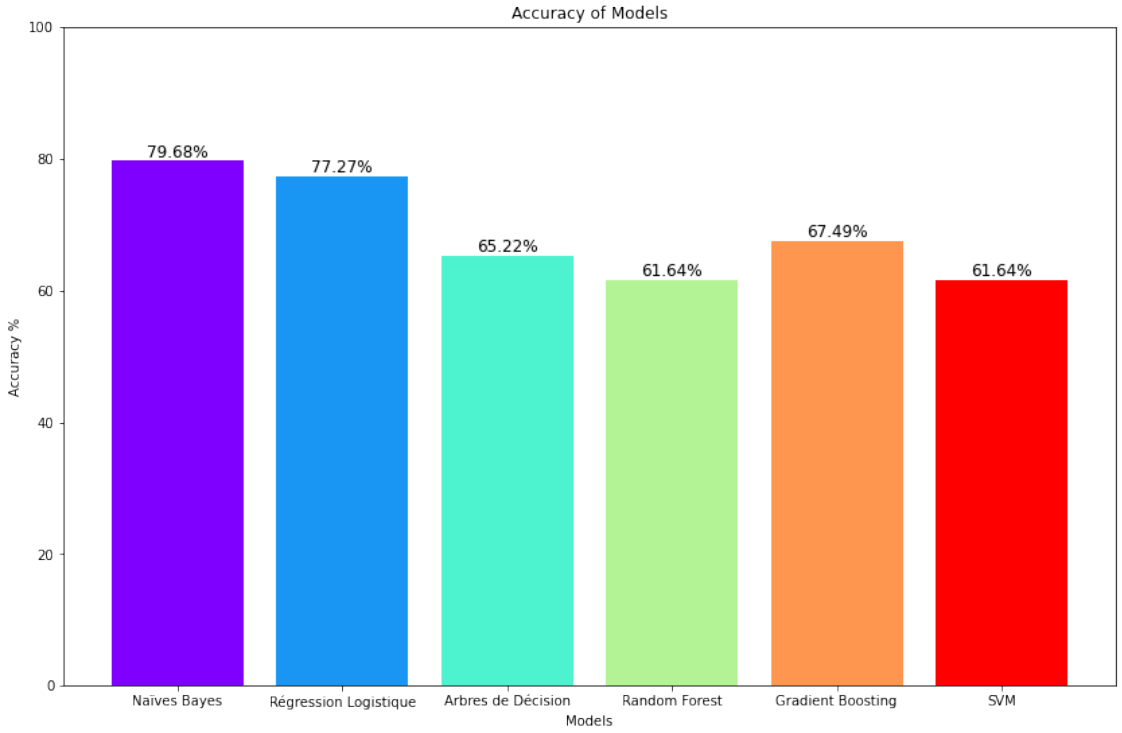
[31]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.cm import rainbow

Models = ['Naïves Bayes', 'Régression Logistique', 'Arbres de Décision', 'Random␣
 ↪Forest', 'Gradient Boosting', 'SVM']
Accuracy = [nb_accuracy*100, lr_accuracy*100, dt_accuracy*100, rf_accuracy*100,␣
 ↪gb_accuracy*100, svm_accuracy*100]

colors = rainbow(np.linspace(0, 1, len(Models)))

plt.figure(figsize=(14,9))
plt.bar(Models, Accuracy, color = colors)
for i in range(len(Models)):
    plt.text(i, Accuracy[i], "{:.2f}%".format(Accuracy[i]), ha='center',␣
 ↪va='bottom' ,size=12)
plt.ylim([0,100])
plt.xlabel('Models')
plt.ylabel('Accuracy %')
plt.title('Accuracy of Models')
```

[31]: `Text(0.5, 1.0, 'Accuracy of Models')`



6 Conclusion

Le **meilleur** modèle est basé sur **Naïves Bayes**. Ce modèle obtient en effet le meilleur taux de classification pour les données de validation avec plus de 79% de bonnes prédictions.

Le **pire** modèle est basé sur **Random Forest** et **SVM**. Ce résultat montre les capacités relativement faibles de prédiction de ce type d'algorithme.

Le meilleur modèle pour ce projet est celui dont les prévisions sont les plus proches des résultats attendus et qui permet le passage à l'échelle: **Naïves Bayes** et **Régression Logistique** sont des bonnes options pour ce projet.

Mais nous pouvons combiner les meilleurs modèles trouvés dans l'étape précédente afin d'améliorer la performance globale de prédiction de ces modèles.